6 Lab: Simple NN (35 pts)

For getting started with deep Neural Network model easily, we consider a simple Neural Network model here and details of the model architecture is given in Table 1. This lab question focuses on building the model in PyTorch and observing the shape of each layer's input, weight and output. Please refer to the **NumPy/PyTorch Tutorial slides** on Canvas and the official documentations if you are unfamiliar with PyTorch syntax. Please finish this lab by completing the `SimpleNN.ipynb` notebook file provided on Canvas. The completed notebook file should be submitted to Gradescope.

Name	Type	Kernel size	depth/units	Activation	Strides
Conv 1	Convolution	5	16	ReLU	1
MaxPool	MaxPool	4	N/A	N/A	2
Conv 2	Convolution	3	16	ReLU	1
MaxPool	MaxPool	3	N/A	N/A	2
Conv 3	Convolution	7	32	ReLU	1
MaxPool	MaxPool	2	N/A	N/A	2
FC1	Fully-connected	N/A	32	ReLU	N/A
FC2	Fully-connected	N/A	10	ReLU	N/A

Table 1: The padding for all three convolution layers is 2. The padding for all three MaxPool layers is 0. A flatten layer is required before FC1 to reshape the feature.

Lab 2 (35 points)

In the notebook, first run through the first two code blocks, then follow the instructions in the following questions to complete each code block and acquire the answers.

(a) (10pt) Complete code block 3 for defining the adapted SimpleNN model. Note that customized `CONV` and `FC` classes are provided in code block 2 to replace the nn.Conv2d and nn.Linear classes in PyTorch respectively. The usage of the customized classes are exactly the same as their PyTorch counterparts, the only difference is that in the customized class the input and output feature maps of the layer will be stored in `self.input` and `self.output` respectively after the forward pass, which will be helpful in question (b). After the code is completed, run through the block and make sure the model forward pass in the end throw no errors. Please copy your code of the completed `SimpleNN` class into the report PDF.

(b) (25pt) Complete the for-loop in code block 4 to print the shape of the input feature map, output feature map and the weight tensor of the 5 convolutional and fully-connected layers when processing a single input. Then compute the number of parameters and the number of MACs in each layer with the shapes you get. In your report, use your results to fill in the blanks in Table 2.

Layer	Input shape	Output shape	Weight shape	# Param	# MAC
Conv 1					
Conv 2					
Conv 3					
FC1					
FC2					

Table 2: Results of Lab 2(b).

Lab 3 (Bonus 10 points)

Please first finish all the required codes in Lab 2, then proceed to code block 5 of the notebook file.

(a) (2pt) Complete the for-loop in code block 5 to plot the histogram of weight elements in each one of the 5 convolutional and fully-connected (FC) layers.

(b) (3pt) In code block 6, complete the code for backward pass, then complete the for-loop to plot the histogram of weight elements' gradients in each one of the 5 convolutional and FC layers.

(c) (5pt) In code block 7, finish the code to set all the weights to 0. Perform forward and backward pass again to get the gradients, and plot the histogram of weight elements' gradients in each one of the 5 convolutional and fully-connected layers. Comparing with the histograms you got in (b), are there any differences? Briefly analyze the cause of the difference, and comment on how will initializing CNN model with zero weights will affect the training process. (Note: The CNN initialization methods will be introduced in Lecture 6.)